OVM to UVM Migration
or There and Back Again, a Consultant’s Tale

Mark Litterick
Verification Consultant
Verilab GmbH, Munich, Germany
Transition from OVM to UVM

- **born UVC**: OVM
- **born OVC**: UVM
- **~Years**: -2, -1, Now, +1
- **Projects**: O-to-U, U-to-O
- **VIP**: Born UVC, Born OVC

Support ongoing OVM, provide OVCs to UVM, enable new UVM projects
Two Stage Evaluation

- **Starting point**
  - clean **OVM-2.1.2** OVCs
  - **no** AVM or URM **legacy**
  - different protocols but **common style** & development team
  - => scripts are more effective, but less general purpose!

- **First attempt – early UVM translation**
  - similar to Mentor Graphics’ Verification Academy flow
  - goal: check for gotcha’s in code, proof of concept, project running in UVM

- **Second attempt – late UVM translation**
  - optimized flow to do most effort in live OVM project
  - automatic and repeatable translation for delivery to UVM
Early UVM Translation

- **Audit**
  - audit OVM code
    - remove deprecated OVM
    - replace non-recommended OVM
  - translate OVM to UVM
    - execute `ovm2uvm` script
    - update regression & test scripts

- **O-to-U**
  - convert to UVM style
    - convert phases to UVM
    - convert *stop to objections
    - convert objection raise/drop to UVM
    - convert set/get to `config_db`
    - convert virtual interfaces to `config_db`
  - use new UVM features
    - remove deprecated UVM
    - use improved UVM message macros
    - use improved UVM command args
    - use UVM register model

- **New**

**Automatic?**
- Find Fix

- **Flow works!**
- **too many changes after translation**

---

since our VIP from live OVM projects
Push Back to OVM

Features already in OVM-2.1.2 source:
- only objection handling used for end-of-test
- improved message macros used instead of methods
- using backported uvm_reg in OVM environments

Key improvements that can be done in OVM:
- virtual interface configuration
- deprecate sequence utilis and OVM sequence library
Example: Interface Configuration

- Improve virtual interface configuration
  - better interface container – specifically to help OVM to UVM translation
  - container set and get methods similar to `uvm_config_db`

```cpp
// example set in OVM testbench module
my_container#(virtual my_if)::set("**", "cif", mif);

// example get in OVM agent or test class
if (!my_container#(virtual my_if)::get(this, "cif", vif))
  `ovm_fatal("NOVIF","..."))
```

```cpp
// example set in UVM testbench module
uvm_config_db#(virtual my_if)::set(null, "**", "cif", mif);

// example get in UVM agent or test class
if (!uvm_config_db#(virtual my_if)::get(this, ",", "cif", vif))
  `uvm_fatal("NOVIF","..."))
```
Example: Deprecated UVM

- **Sequencer & sequence utils** deprecated in UVM
  - OVM sequence library **not required** in UVM or OVM
  - automatic script & manual repair (for reactive slaves)

```verilog
class my_seq extends ovm_sequence #(my_seq_item);
  `ovm_object_utils(my_seq)
  `ovm_sequence_utils(my_seq, my_seq)
  `ovm_declare_p_sequencer(my_seq)

class my_sequencer extends ovm_sequence #(my_seq_item);
  `ovm_sequence_utils(my_sequencer)
  `ovm_update_sequence_lib_and_item(my_seq_item)
  `ovm_component_utils(my_sequencer)

class my_env extends ovm_env;
  set_config_int("*.my_sequencer", "count", 0);
  set_config_string("*.my_sequencer","default_sequence","my_seq");
  `ovm_do_on(my_seq, my_env.my_sequencer)
  my_seq.start(my_env.my_sequencer);
```

*can be done in OVM or UVM*

*do once in OVM*
Late UVM Translation

**Audit**
- audit OVM code
  - remove deprecated OVM ✓ x
  - delete non-recommended OVM ✓ x
  - convert *stop to objections ✓ x
  - convert virtual interfaces to container ✓ x
  - remove deprecated OVM seq* utils ✓ ✓
  - use improved OVM message macros ✓ ✓
  - use UVM register model (back-ported) ✓

**O-to-U**
- translate OVM to UVM
  - execute ovm2uvm script ✓ ✓

**Convert**
- convert to UVM style
  - convert phases to UVM ✓ ✓
  - convert objection raise/drop to UVM ✓ ✓
  - convert set/get to config_db ✓ ✓
  - convert virtual interfaces to config_db ✓ ✓

**New**
- use new UVM features
  - use improved UVM command args ✓ x
Final Translation Process

- Prepare source OVC for translation – once
- Continue development of OVC in live OVM project
- Release OVC versions to UVM when appropriate
- Automatic translate to UVM as part of VIP release

<table>
<thead>
<tr>
<th>OVM</th>
<th>UVM</th>
<th>Automatic?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit</td>
<td>audit OVM code</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>O-to-U</td>
<td>• execute audit script</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Convert</td>
<td>translate OVM to UVM</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>New</td>
<td>• execute ovm2uvm script</td>
<td>✓ ✓</td>
</tr>
<tr>
<td></td>
<td>• convert to UVM style</td>
<td>✓ ✓</td>
</tr>
<tr>
<td></td>
<td>• use new UVM features</td>
<td>✓ ✗</td>
</tr>
<tr>
<td></td>
<td>• use improved UVM command args</td>
<td>✓ ✗</td>
</tr>
</tbody>
</table>
UVM to OVM Back-Porting

- Slim OVM to UVM conversion supports reverse translation
- Valid when UVM transition period expected to endure
- Translate new UVC to OVC for ongoing OVM projects
- **UVM limitations (hard to back-port)**
  - avoid run-time phases
  - avoid TLM-2.0
- **Other considerations (easier to back-port)**
  - modified objection handling
  - updated phase methods
  - config_db changes
  - command line processor

still no industry consensus
use sequence-based phasing
localize TLM2 if really required
OK if no run-time phases
normally OK
goal is not to cripple UVM but enable reuse in OVM
Conclusion

- **Goal is move to UVM**
  - transition period could endure for some time
  - considerable **OVM legacy** and many ongoing projects
  - new UVM projects need **OVC** libraries
  - ongoing OVM projects may need new **UVCs**

- **Presented an overview of migration process**
  - prepare OVM for easier translation
  - slim automatic translation process
  - translation process is **reversible**

- **Developed on family of OVCs, several projects**
  - applied to multiple projects @ different clients